Qualified: Assessment Result

# Steve Geers

## **General Programming** Results

### Candidate Remarks

I would have preferred that the functions pass a target variable for the function output instead of forcing me to use global strings.

### Assessment Summary

100%

Team

Global

**1 h, 54 m, 54 s** Active Time

10th Team Percentile

10th Global Percentile

**Invited** by **Dev Eval** on Friday, September 4, 2020 4:00 PM
Sent 2 times, and opened 6 times

Steve Geers **opened** this assessment on Saturday, September 5, 2020 11:38 AM

Steve Geers **started** this assessment on Saturday, September 5, 2020 11:39 AM

Steve Geers **submitted** this assessment after **11 hours and 24 minutes** on Saturday, September 5, 2020 11:03 PM

This candidate spent **1 hour and 54 minutes** active in the browser working on the assessment which places them in the **10th Team Percentile** amongst candidates

### Review Summary

Danny Brinlee –

### Solutions Summary

| Challenge | Score | | Active Time |
|---|---|---|---|
| C #1: Get the Middle Character | 100% | Team | 11 m, 19 s |
| C #2: Multiples of 3 and 5 | 100% | Team | 8 m, 22 s |
| C #3: Weight for weight | 100% | Team | 22 m, 17 s |

| Challenge | Score | Active Time |
| --- | --- | --- |
| C #4: Base64 Encoding | 100% Team | 1 h, 12 m, 56 s |

Qualified: Assessment Code Solution

# #1: Get the Middle Character

## Scoring

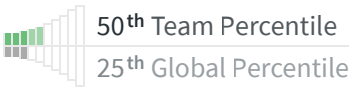
100%

4 / 4 Tests (2 Attempts)


## Timing

**11 m, 19 s** Active Time

**912** ms Run Time




## Max Code Similarity

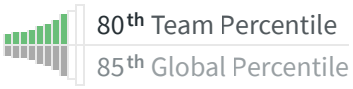
64%

2 Matches


## Code Complexity

**67**

Lower is better




## Challenge Reviews

**Danny Brinlee**

## Instructions

You are going to be given a word. Your job is to return the middle character of the word. If the word's length is odd, return the middle character. If the word's length is even, return the middle 2 characters.

### `get_middle(str)`

Find the middle character(s) of a word.

**Parameters**

`str`: ***char**** - word to pull middle characters from

**Return Value**

***char**** - letter(s) in the middle of the word

**Constraints**

`0 < str < 1000`

Examples

| str | Return Value |
|---|---|
| `"test"` | `"es"` |

| str | Return Value |
| --- | --- |
| `"testing"` | `"t"` |
| `"middle"` | `"dd"` |

## Solution Code

C

```c
#include <stdio.h>
#include <string.h>

#define MAX_LEN 100
//Global variables needed for char *get_middle()
/*
This function returns a pointer to the middle charactre if the
String length  is odd and the middle two characters if the
stringlength is even

 This function reuqires a globl varible since cosntant
 strings can be passed to it
 */
char gblStr[3];
char *get_middle( char* str )

{
    int length;
    //determine the length
    length  = strlen(str);
    if(1 == length%2)
    {// odd length
        gblStr[0] = str[length/2];
        gblStr[1] = 0;

    }
    else
    { //even length
        gblStr[0] = str[length/2-1];
        gblStr[1] = str[length/2];
        gblStr[2] = 0;
    }
    return gblStr;
}

```

## Candidate's Tests

```
#include <criterion/criterion.h>
char* get_middle( char* str );
Test(get_middle, should_find_the_middle_characters)
{
  char* actual0 = get_middle("A");
  char* expected0 = "A";
  cr_assert_str_eq(actual0, expected0);
  char* actual1 = get_middle("test");
  char* expected1 = "es";
  cr_assert_str_eq(actual1, expected1);
  char* actual2 = get_middle("testing");
  char* expected2 = "t";
  cr_assert_str_eq(actual2, expected2);
  char* actual3 = get_middle("middle");
  char* expected3 = "dd";
  cr_assert_str_eq(actual3, expected3);
};
```

Qualified: Assessment Code Solution

# #2: Multiples of 3 and 5

### Scoring

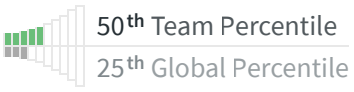
100%

4 / 4 Tests (1 Attempt)


### Timing

**8 m, 22 s** Active Time

**1,073** ms Run Time




### Max Code Similarity

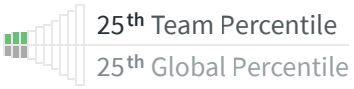
70%

1 Match


### Code Complexity

**42**

Lower is better




## Challenge Reviews

**Danny Brinlee**

## Instructions

## Task

If we list all the natural numbers below 10 that are multiples of 3 or 5, we get 3, 5, 6 and 9. The sum of these multiples is 23.

Complete the function `solution` so that it returns the sum of all the multiples of 3 or 5 **below** the number passed in.

# Specification

`solution(number)`

Finds the sum of multiples of 3 or 5 that is less than the provided number

**Parameters**

`number`: ***int*** - Maximum number to check against

**Return Value**

***int*** - Sum of all multiples of either 3 or 5

Examples

| number | Return Value |
|---|---|

| number | Return Value |
| --- | --- |
| `10` | `23` |
| `200` | `9168` |

## Solution Code

C

```c
/*
    returns the sum of all the numbers less than the number that are multiples of 3 or 5
*/
int solution(int number) {
    int sum;
    int i;
    sum = 0;
    for (i = 0; i < number; i++){

        if (0 == i%3){ // it is a multiple of 3
            sum += i;
            continue;
        }

        if (0 == i%5){ //it is a multiple of 5
            sum += i;
            continue;
        }
    }
    return sum;
}
```

## Candidate's Tests

```c
#include <criterion/criterion.h>

int solution(int number);

Test(solution, should_handle_the_example) {
    int actual = solution(10);
    int expected = 23;
    cr_assert_eq(actual, expected,
                 "expected %d but got %d", expected, actual);
};
```

Qualified: Assessment Code Solution

# #3: Weight for weight

### Scoring

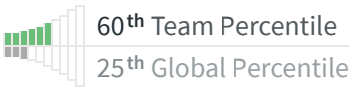
100%

4 / 4 Tests (7 Attempts)


### Timing

**22 m, 17 s** Active Time

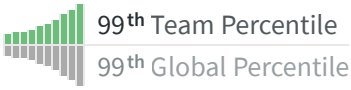
**991** ms Run Time




### Code Complexity

**271**

Lower is better




## Challenge Reviews

**Danny Brinlee**

### Candidate Notes:

This solution is based on these assumptions:
The number of entries is less than MAX_ENTRIES
The average number of digits is less than 4.

This was done to avoid using malloc() or any of its vaiants.

### Instructions

You are provided a string containing a list of positive integers separated by a space ( `" "` ). Take each value and calculate the sum of its digits, which we call it's "weight". Then return the list in ascending order by weight, as a string joined by a space.

For example `99` will have "weight" `18` , `100` will have "weight" `1` so in the ouput `100` will come before `99` .

## Specification

`order_weight(strng)`

**Parameters**

`strng`: ***char**** - String of digits to be summed and put in order

**Return Value**

***char**** - A string of digits ordered by their "weight"

**Example:**

`"56 65 74 100 99 68 86 180 90"` ordered by numbers weights becomes:
`"100 180 90 56 65 74 68 86 99"`

When two numbers have the same "weight", let's consider them to be **strings** and not numbers:

`100` is before `180` because its "weight" ( `1` ) is less than the one of `180` ( `9` )
and `180` is before `90` since, having the same "weight" ( `9` ) it comes before as a **string**.

All numbers in the list are positive integers and the list can be empty.

## Solution Code

C **C**

```c
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdint.h>

#define MAX_ENTRIES 1000
#define NEEDED_STR_LEN MAX_ENTRIES*4+1

char gblStr[MAX_ENTRIES * 4+1];

int compareFunction(const void * elem1, const void * elem2);

typedef struct{
    int64_t number;
    int64_t sum;
} numList;

/*
 * Assumes the the number of entire is less than MAX_ENTRIES
 * to avoid using malloc
 *
 */

char* order_weight( char* str )
{
    numList theList[MAX_ENTRIES];
    int fieldDone;
    int strDone;
    int count;
    int64_t currentNum;
    int64_t currentSum;
    int numFields;
    int fieldCount;
    int length;
    int haveAnumber;

```

```
    numFields = 0;
    count = 0;
    strDone = 0;

    while(0 == strDone){
        fieldDone = 0;
        currentNum = 0;
        currentSum = 0;
        haveAnumber = 0;

        while(0 == fieldDone){
            if(isdigit( str[count])){// process digits
                currentNum  *= 10LL;
                currentNum += (int64_t)(str[count] - 0x30);
                currentSum += (int64_t)(str[count] - 0x30);
                haveAnumber = 1;
                count++;
                continue;
            }else{
                fieldDone = 1;
                if(0 == str[count]){// end of string
                    strDone = 1;
                }
                if(1 == haveAnumber){//only do if have a number
                    theList[numFields].sum = currentSum;
                    theList[numFields].number = currentNum;
                    numFields +=1;
                    count++;
                }
            }

        }

    }

    //Sort the numbers
    qsort(theList, numFields, sizeof(numList), compareFunction);

    gblStr[0] = 0; //clear the string


    for(fieldCount = 0; fieldCount < numFields; fieldCount++){
        int outLen = strlen(gblStr);
        // need to append, not overwrite
        sprintf(&gblStr[outLen], "%ld ",  theList[fieldCount].number);
    }


    length = strlen(gblStr);
    gblStr[length-1] = 0; // remove the last space

    return gblStr;
}

```

```
/*
 * comparison function for qsort
 sorts nmber based on the sum member
 */
int compareFunction(const void * elem1, const void * elem2)
{
    numList first = *((numList*)elem1);
    numList second = *((numList*)elem2);

    if (first.sum > second.sum){
        return  1;
    }

    if (first.sum < second.sum){
       return -1;
    }

    char str1[22];
    char str2[22];
    sprintf(str1,"%ld", first.number );
    sprintf(str2,"%ld", second.number );
    int compareVal = strcmp(str1, str2);

    //return 0;
    return compareVal;
}

```

## Candidate's Tests

```
#include <criterion/criterion.h>
char* order_weight( char* str );
Test(order_weight, should_handle_the_example)
{
  char* actual0 = order_weight("56 65 74 100 99 68 86 180 90");
  char* expected0 = "100 180 90 56 65 74 68 86 99";
  cr_assert_str_eq(actual0, expected0);
};
```

Qualified: Assessment Code Solution

# #4: Base64 Encoding

## Scoring

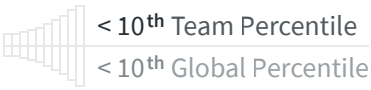
100%

3 / 3 Tests (3 Attempts)


## Timing

**1 h, 12 m, 56 s** Active Time

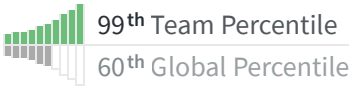
**982** ms Run Time




## Code Complexity

**230**

Lower is better




## Challenge Reviews

**Danny Brinlee**

### Instructions

Create a function or a method that converts the value of the String **to Base64** using the ASCII (UTF-8 for C#) character set.

Do not use built in functions.

As an example the input string `this is a string!!` should produce the output string `dGhpcyBpcyBhIHN0cmluZyEh`.

You can learn more about Base64 encoding and decoding here (http://en.wikipedia.org/wiki/Base64) .

# Specification

`to_base64(str)`

**Parameters**

`str`: ***char**** - String to be converted using Base64 format

**Return Value**

***char**** - The converted string

Examples

| str | Return Value |
|---|---|
| `"this is a string!!"` | `"dGhpcyBpcyBhIHN0cmluZyEh"` |

| str | Return Value |
| --- | --- |
| `"ABCDEFGHIJKLMNOPQRSTUVWXYZ "` | `"QUJDREVGR0hJSktMTU5PUFFSU1RVVldYWVog"` |

## Solution Code

C

```c
#include <stdint.h>
#include <stdint.h>
#include <string.h>

char gblStr[100];

char r64lookup[64] = {'A','B','C','D','E','F','G','H',\
                      'I','J','K','L','M','N','O','P',\
                      'Q','R','S','T','U','V','W','X',\
                      'Y','Z','a','b','c','d','e','f',\
                      'g','h','i','j','k','l','m','n',\
                      'o','p','q','r','s','t','u','v',\
                      'w','x','y','z','0','1','2','3',\
                      '4','5','6','7','8','9','+','/'};

void to_radix64( char* outstr, char* instr );

char* to_base64( char* str ){
    char fourBytesOut[5];
    char threeBytesIn[4];
    int count;
    int numBlocks ;

    int wholeLen = strlen(str);

    numBlocks = 0;
    gblStr[0] = 0;

    while (numBlocks*3 < wholeLen){
        //get 3 bytes
        for(count = 0; count<3; count++){
            threeBytesIn[count] = str[count + numBlocks * 3];
        }
        threeBytesIn[4] = 0;
        to_radix64(fourBytesOut, threeBytesIn);
        strcat(gblStr, fourBytesOut);
        numBlocks +=1;
    }

    return gblStr;
}

```

```
/*
 * convers a full 3 byte char str to a 4 byte fadix64 string
 */
void to_radix64( char* outstr, char* instr )
{
    uint32_t blockSum;
    uint32_t mask;
    uint8_t byte;
    uint8_t count;

    //create sum of the 3 bytes to be converted to a4 byte radix64 number
    blockSum = ((uint32_t)instr[0])<<16;
    blockSum += (((uint32_t)instr[1]) <<8);
    blockSum += ((uint32_t)instr[2]);

    // pull off 6 bits and put in radix64 fr
    mask = 0x0000003f;
    for(count = 0; count < 4; count++){
        byte = (uint8_t)(blockSum & mask);
        outstr[3-count] = r64lookup[byte];
        blockSum = blockSum >>6;
    }

    outstr[4] = 0;//null terminate the sting
}

```

## Candidate's Tests

```
#include <criterion/criterion.h>
char* to_base64( char* str );
Test(to_base64, should_handle_sentences)
{
  char* actual0 = to_base64("this is a string!!");
  char* expected0 = "dGhpcyBpcyBhIHN0cmluZyEh";
  cr_assert_str_eq(actual0, expected0);
  char* actual1 = to_base64("this is a test!");
  char* expected1 = "dGhpcyBpcyBhIHRlc3Qh";
  cr_assert_str_eq(actual1, expected1);
};
```